Journal of Biomedical Informatics 55 (2015) 41–54

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Adaptive semi-supervised recursive tree partitioning: The ART towards large scale patient indexing in personalized healthcare

Fei Wang

Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA

ARTICLE INFO

Article history:
Received 15 October 2014
Accepted 21 January 2015
Available online 3 February 2015

Keywords:
Large scale
Indexing
Patients
Tree partitioning
Semi-supervised learning

ABSTRACT

With the rapid development of information technologies, tremendous amount of data became readily available in various application domains. This *big data* era presents challenges to many conventional data analytics research directions including data capture, storage, search, sharing, analysis, and visualization. It is no surprise to see that the success of next-generation healthcare systems heavily relies on the effective utilization of gigantic amounts of medical data. The ability of analyzing big data in modern healthcare systems plays a vital role in the improvement of the quality of care delivery.

Specifically, *patient similarity evaluation* aims at estimating the clinical affinity and diagnostic proximity of patients. As one of the successful data driven techniques adopted in healthcare systems, patient similarity evaluation plays a fundamental role in many healthcare research areas such as prognosis, risk assessment, and comparative effectiveness analysis. However, existing algorithms for patient similarity evaluation are inefficient in handling massive patient data. In this paper, we propose an Adaptive Semi-Supervised Recursive Tree Partitioning (ART) framework for large scale patient indexing such that the patients with similar clinical or diagnostic patterns can be correctly and efficiently retrieved. The framework is designed for semi-supervised settings since it is crucial to leverage experts' supervision knowledge in medical scenario, which are fairly limited compared to the available data. Starting from the proposed ART framework, we will discuss several specific instantiations and validate them on both benchmark and real world healthcare data. Our results show that with the ART framework, the patients can be efficiently and effectively indexed in the sense that (1) similarity patients can be retrieved in a very short time; (2) the retrieval performance can beat the state-of-the art indexing methods.

1. Introduction

With the rapid development of information technologies, *big data* [8] has been one of the major themes in modern data mining research. It presents many challenges to existing computational technologies including data mining, machine learning, database, statistics and information visualization. Similarly, in medical research, the *Electronic Medical Records* (EMR) has been everywhere in hospitals and clinical institutions. Hence, in medical informatics research, there is an emerging need to efficiently utilize a huge number of EMRs to provide effective support and improve the quality of service at the point of care.

Personalization is another popular trend in the current computational research. What personalization emphasizes is to customize the information of interest and tailor it to each individual. Under this context, the goal of *personalized healthcare* is to improve the safety, quality and effectiveness of healthcare for every patient. To make personalized healthcare successful, one key point is to identify the current status of the specific patient so that we can enable the medications and treatments to be tailored to each person's needs. However, in the real world, the exact "status" of a patient would be very difficult to determine as the patient may have a lot of complicated comorbidities. To get around this, an alternative way is to evaluate the *clinical similarity* of patients so that we can make personalized healthcare plans for focus patients by utilizing the successful experience on their cohorts of similar patient cohorts. This strategy has been validated as *collaborative filtering* [13] in information recommendation systems.

There is quite some existing work on effective evaluation of clinical patient similarities. For example, Sun et al. [14] applied a locally supervised metric learning algorithm to evaluate the patient similarities. Wang et al. [20] further extended it so that the metric can be updated interactively to incorporate the physician's feedback fast, and they also make the algorithm work in a heterogeneous scenario [19] where multiple types of feedback (from physicians with different specialties) can be leveraged.

E-mail address: fei_wang@uconn.edu

http://dx.doi.org/10.1016/j.jbi.2015.01.009
1532-0464/

However, the following issues may limit the applicability of the aforementioned methods in real world scenarios:

- Most of the existing approaches are built under the *supervised* setting, where abundant labeled data is desired for learning the metric. However, it is well known that annotation of EMRs requires sophisticated medical professionals, which is very time-consuming and expensive.
- Those approaches require computation of nearest neighbors in a predefined metric space, e.g. Euclidean distance space, for each patient. This procedure is expensive since the EMRs are often represented by high-dimensional feature vectors and the size of the patient population could be very large.

In order to (1) effectively evaluate the pairwise clinical similarities among patients; (2) efficiently retrieve similar patients for any query patient, we propose an Adaptive Semi-Supervised Recursive Tree Partitioning (ART) framework for large scale patient indexing. It is worthwhile to highlight the following aspects of the proposed approach:

- *Semi-supervised*. At each node of the indexing tree, our ART framework learns a projection direction by solving some optimization problem to partition the data. The objective of such optimization problem is composed of two terms, a *supervised* term containing the expertise knowledge, and an *unsupervised* term measuring certain data properties such as the variance in the projected space. The tradeoff of these two terms is also adaptive in the sense that if there are more unlabeled data contained in one node, the unsupervised term will receive more weight, resulting in a more balanced partitioning. However, if the node contains fewer points, the data properties will not rely that much and the supervised term will receive larger weight, resulting more accurate partitioning. In this way we aim to construct a *balanced* yet *accurate* indexing tree.
- *Adaptive indexing*. As learning exact pairwise distances may suffering from huge computational burden, we will develop smart indexing approaches for approximate nearest neighbor search, where rather than using a uniform distance measure for all the patients in the indexing tree as in kd-tree [1], we learn a specific distance metric for a subset of patient population within a single node based on the data distribution. In this way, we tend to construct a more *accurate* indexing tree for retrieving similar patients.

Based on the proposed ART framework, we implemented several instantiations using different measurements of the supervised and unsupervised components. Empirical study on both benchmark and real world patient data clearly demonstrate the superiority of the ART based approaches over traditional indexing methods. Such framework has great potential on physician decision support, such as prognosis and disease early prediction.

The rest of this paper is organized as follows. Section 2 will review some related works including hashing and indexing. The detailed ART framework as well as several of its instantiations are introduced in Section 3. The experiments on two benchmark data sets are introduced in Section 4, followed by the experimental results on real world data in Section 5 and conclusions in Section 6.

2. Related works

Patient similarity evaluation is a domain specific case of the generic similarity based nearest neighbors (NN) search, which has been identified as a fundamental problem in many data mining algorithms and practical information process systems, ranging from graph learning to classification [15,16,25] to information retrieval [12,17,21]. An intuitive way to identify the most similar samples is to perform exhaustive comparison. For instance, given a query patient $\mathbf{q}$ and a similarity function $sim(\cdot)$, the nearest patient $\mathbf{x}^*$ in a database with n data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ is given by $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} sim(\mathbf{q}, \mathbf{x})$. Exhaustive search requires linear computational cost $\mathcal{O}(n)$, which is infeasible for large scale and high dimensional applications due to time and space constraints.

Instead of finding the exact nearest neighbors, recently the research community tends to design efficient indexing techniques to search *Approximate Nearest Neighbors* (ANN). Typically, the ANN techniques require much less search time, such as those with the time complexity as sublinear $o(n)$, logarithmic $\mathcal{O}(\log n)$, or even constant $\mathcal{O}(1)$. In addition, sometimes certain performance guarantee like ϵ-NN property can be provided also [3]. Briefly, two popular indexing methods, i.e. tree-based and hashing-based are studied for general ANN search tasks. Both of these two types of methods rely on certain partitioning strategy of the sample space, but with different strength and uniqueness.

Hashing based approaches explore *repeated partitioning* on the dataset and generate one-bit binary codes for all data points after each single partitioning. As long as the data can be indexed by binary hash codes, it requires sublinear or even constant time to perform hash table lookup. Linear projection based hashing methods such as the locality sensitive hashing (LSH) are representative hashing approaches, which perform data partitioning using random projections and thresholds [3]. Although LSH has theoretic guarantees for certain distance metric spaces, it is often practically inefficient or inaccurate [21]. Recently many machine learning techniques have been leveraged to design more efficient data-dependent hash functions, such as semi-supervised hashing [21], binary reconstructive embedding based hashing [6] and sequential hashing [22].

Although hashing methods are extremely scalable, the indexing accuracy has been observed to be limited. In our application for retrieving similar patients, a typical patient dataset may not be that large (e.g., billions). In this case, tree-based indexing can be easily performed in a regular modern computer system. Different with hashing methods, tree-based techniques perform *recursive partitioning* on the data and represent the indexing by a tree structure. The search efficiency of tree-based significantly outperforms exhaustive search and usually only needs logarithmic time. The search accuracy of tree-based indexing is usually superior and even can retrieve the exact nearest neighbors through some backtracking strategy [7]. Several representative tree-based methods include the well-known kd-tree [1], ball tree [10], and metric tree [18]. However the existing tree methods are mostly unsupervised without considering to use the available label information. Also as discussed earlier, tree-based methods are also constrained by certain metric space like Euclidian space. To tackle such limitations, in the following, we present an adaptive semi-supervised recursive tree partitioning method and show that the proposed tree indexing can be used to efficiently search similar patients under the scenario of the medical domain.

3. Methodology

Suppose we have a set of patients $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the profile of the ith patient and d is the dimensionality of the patient vector. We use $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ to represent the entire patient matrix. Our goal is to build a tree structure to index these patient profiles so that the nearest neighbors of a query patient $\mathbf{q}$ can be rapidly retrieved. The element on each dimension of a patient profile represents the value of a specific medical events, for example, the frequency of a diagnosis code over a

certain time period. Besides those profiles, we also assume that there is some expertise knowledge available, which is in the form of pairwise constraints on $\mathcal{X}$. For instance, if patient i and j are similar to each other, then we put a must-link between $\mathbf{x}_i$ and $\mathbf{x}_j$. Similarly, if patient i and j are dissimilar to each other, then we put a cannot-link between them. We construct a must-link set $\mathcal{M}$ by collecting all patient pairs with must-links, and a cannot-link set $\mathcal{C}$ by collecting all patient pairs with cannot-links. In practice, $\mathcal{M}$ and $\mathcal{C}$ are coming from experts' knowledge, i.e., the physician provides them. Assume we have a total of l patients with pairwise labels represented as $\mathbf{X}_l \in \mathbb{R}^{d\times l}$, where each patient in $\mathbf{X}_l$ is involved in at least one must- or cannot-link. In the following presentation we will refer the patient profiles as data, and $\mathbf{X}$ as the data matrix, $\mathbf{x}_i$ as the i-th data vector.

Similar as traditional tree indexing algorithms, we will construct binary space partitioning trees. From the root, the data points in $\mathcal{X}$ are split into two halves by a partition hyperplane, and each half is assigned to one child node. Then each child node is recursively split in the same manner to create the tree. At each node, we find the partition hyperplane $\mathbf{w}$ by optimizing the following objective

$$\mathcal{J}(\mathbf{w}) = \mathcal{J}_{\mathcal{S}}(\mathbf{w}) + \lambda \mathcal{J}_{\mathcal{U}}(\mathbf{w}) \tag{1}$$

where $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ is some supervised term involving expert knowledge encoded in $\mathcal{M}$ and $\mathcal{C}$, and $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ is a pure data-dependent term without integrating any medical supervision. The constant λ is a tradeoff parameter, which balances the contributions from both terms. In the following, we will discuss several different choices of those two terms with different uniqueness and emphasis.

3.1. The choice of $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$

In general, there are two typical principles on the construction of $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$. One is to maximize the data variance after projection, and the other is to maximally separate the data clusters in its intrinsic space. Below is the detailed descriptions for constructing $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ using different principles.

3.1.1. Variance maximization

This criterion has been commonly adopted in metric trees. The goal of this type of approach is to find the direction under which the variance of the projected data is maximized, such that the binary partition on those directions will more likely to produce a balanced tree. Therefore the constructed tree will not be that deep so the nearest neighbors of a query data point can be quickly found. One common ways to achieve this goal is to apply *Principal Component Analysis* (PCA) [5]. This method obtains the eigenvector from the data covariance matrix with the largest corresponding eigenvector, which means we need to maximize the following objective

$$\mathcal{J}_{\mathcal{U}_{PCA}}(\mathbf{w}) = \frac{1}{n^2}\mathbf{w}^\top \bar{\mathbf{X}}\bar{\mathbf{X}}^\top \mathbf{w} = \frac{1}{n}\mathbf{w}^\top \mathbf{X}\left(\mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^\top\right)\mathbf{X}^\top \mathbf{w} \tag{2}$$

where $\bar{\mathbf{X}} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_n]$ is the centered data matrix with $\bar{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j$, $\mathbf{I}$ is the order n identity matrix, and $\mathbf{e}$ is the n-dimensional all-one vector.

3.1.2. Cluster separation

Different from variance maximization based approaches, cluster separation based methods tend to seek a projection direction under which the two balanced data clusters can be formed and equally distributed on the two sides of the median of the projected data. Following the idea in spectral clustering [9], the data clusters can be obtained by maximizing the following objective

$$\mathcal{J}_{\mathcal{U}_{CS}}(\mathbf{w}) = -\sum_{\mathbf{x}_i\in\mathcal{G}_1, \mathbf{x}_i\in\mathcal{G}_2} w_{ij} = -\mathbf{w}^\top \mathbf{X}(\mathbf{D} - \mathbf{W})\mathbf{X}^\top \mathbf{w} \tag{3}$$

where $\mathcal{G}_1$ and $\mathcal{G}_2$ are the two data clusters. $\mathbf{W} \in \mathbb{R}^{n\times n}$ is the data similarity matrix with its (i,j)th entry is computed as

$$W_{ij} = \exp(-\alpha\|x_i - \mathbf{x}_j\|^2) \tag{4}$$

$\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$. One thing we want to emphasize here is that here we want to minimize $\mathcal{J}_{\mathcal{U}_{CS}}(\mathbf{w})$, instead of maximize. One bottleneck for constructing $\mathcal{J}_{\mathcal{U}_{CS}}(\mathbf{w})$ is that we need to compute an $n \times n$ data similarity matrix $\mathbf{W}$ as well as do matrix multiplication on $\mathbf{X}$ and $\mathbf{D} - \mathbf{W}$, which could be very time consuming when n is huge.

3.2. The choice of $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$

As we mentioned earlier, the construction of $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ should incorporate experts' supervision information on the data, i.e., leveraging the knowledge contained in $\mathcal{M}$ and $\mathcal{C}$. Below we will also present two options for constructing $\mathcal{J}_{\mathcal{S}}(\mathbf{W})$.

3.2.1. Projection perspective

This perspective treats $\mathbf{w}$ as a pure projection that maps the data onto a one-dimensional space. After the projection, we want the data pairs in $\mathcal{M}$ to be distributed as compact as possible, while the data pairs in $\mathcal{C}$ to be distributed as scattered as possible. One straightforward criterion is to minimize the overall pairwise distances for the data pairs in $\mathcal{M}$ while maximizing the overall distances for the data pairs in $\mathcal{C}$, i.e., maximizing the following objective

$$\begin{aligned}\mathcal{J}_{\mathcal{S}_{proj}} &= \frac{1}{|\mathcal{C}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}}(\mathbf{w}^\top\mathbf{x}_i - \mathbf{w}^\top\mathbf{x}_j)^2 - \frac{1}{|\mathcal{M}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}}(\mathbf{w}^\top\mathbf{x}_i - \mathbf{w}^\top\mathbf{x}_j)^2 \\ &= \sum_{ij}(\mathbf{w}^\top\mathbf{x}_i - \mathbf{w}^\top\mathbf{x}_j)^2 S_{ij} = \mathbf{w}^\top\mathbf{X}_l(\mathbf{E} - \mathbf{S})\mathbf{X}_l^\top\mathbf{w}\end{aligned} \tag{5}$$

where $\mathbf{S}$ is an $l \times l$ matrix and its (i,j)th entry is defined as

$$S_{ij} = \begin{cases} -\frac{1}{|\mathcal{M}|}, & \text{if } (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M} \\ \frac{1}{|\mathcal{C}|}, & \text{if } (\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C} \\ 0, & \text{otherwise}\end{cases} \tag{6}$$

and $|\cdot|$ denotes the cardinality of a set. $\mathbf{E}$ is an $l \times l$ diagonal matrix with $E_{ii} = \sum_j S_{ij}$.

3.2.2. Prediction perspective

This type of approach treats the projection as a linear prediction function $f(\mathbf{x})$ such that the sign of $f(\mathbf{x})$ indicates the class of $\mathbf{x}$. If we assume that the data in each node are centralized, then we can neglect the bias b in $f(\mathbf{x})$ such that $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}$. The supervised term $\mathcal{J}_{\mathcal{S}}$ under the prediction perspective is defined as

$$\begin{aligned}\mathcal{J}_{\mathcal{S}_{pred}} &= frac1|\mathcal{C}|\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}}\mathbf{w}^\top\mathbf{x}_i\mathbf{x}_j^\top\mathbf{w} - \frac{1}{|\mathcal{M}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}}\mathbf{w}^\top\mathbf{x}_i\mathbf{x}_j^\top\mathbf{w} \\ &= \mathbf{w}^\top\left(\frac{1}{|\mathcal{C}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}}\mathbf{x}_i\mathbf{x}_j^\top - \frac{1}{|\mathcal{M}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}}\mathbf{x}_i\mathbf{x}_j^\top\right)\mathbf{w} \\ &= \mathbf{w}^\top\mathbf{X}_l\mathbf{S}\mathbf{X}_l^\top\mathbf{w}\end{aligned} \tag{7}$$

where $\mathbf{S} \in \mathbb{R}^{l\times l}$ is a symmetric matrix with its (i,j)th entry defined in Eq. (6). With any specific combinations of $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ and $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ mentioned above, we can construct a concrete form of $\mathcal{J}$ as in Eq. (1). It is not difficult to observe that any combinations of $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ and

$\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ introduced in this section will result in a $\mathcal{J}$ in the form of $\mathbf{w}^\top\mathbf{X}\mathbf{A}\mathbf{X}^\top\mathbf{w}$ with different $\mathbf{A}$ matrix.

3.3. Optimization

From those descriptions above we can see that no matter what the choices of $\mathcal{J}_{\mathcal{S}}$ and $\mathcal{J}_{\mathcal{U}}$ are, the final objective $\mathcal{J}$ can always be rewritten in the form of $\mathcal{J} = \mathbf{w}^\top\mathbf{A}\mathbf{w}$. The matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ is some matrix having different concrete forms depending on the specific choices of both supervised and unsupervised components. In this case, if we further constrain $\mathbf{w}$ to have unit norm, then the optimization problem we want to solve at each node becomes[1]

$$\max_{\mathbf{w}:\mathbf{w}^\top\mathbf{w}=1} \mathbf{w}^\top\mathbf{X}\mathbf{A}\mathbf{X}^\top\mathbf{w} \tag{8}$$

This becomes a standard *Rayleigh Quotient* optimization problem [4], and the optimal solution $\mathbf{w}^*$ can be obtained as the eigenvector of $\mathbf{X}\mathbf{A}\mathbf{X}^\top$ whose corresponding eigenvalue is the largest.

3.4. Complexity analysis

In this subsection, we analyze the space and computational complexities of the proposed Adaptive Recursive Tree (ART) partitioning tree methods. For space complexity, as we need to store the data median for each node for partitioning, which requires $\mathcal{O}(n)$ space. Moreover, additional storage is required to store the projection for each node, resulting in the complexity $\mathcal{O}(dn)$. Thus the complete space complexity for ART series methods is $\mathcal{O}((d+1)n)$.

The computational cost of the ART methods mainly lies in the following aspects: (1) construct matrix $\mathbf{A}$ over the entire dataset, (2) extracting the projections by eigen-decomposition, and (3) compute the projected points. Here we omit the time cost for calculating $\mathcal{J}_s$ since we assume $l \ll n$ for general semi-supervised settings. Specifically, it takes $\mathcal{O}(nd^2)$ to compute the data covariance matrix (as in Section 3.1.1) or $\mathcal{O}(n^2d)$ to compute data similarity matrix (as in Section 3.1.2). To decompose a matrix $\mathcal{J}$ with the size $d \times d$ to derive the principal projections, it requires $\mathcal{O}(d^3)$. Finally, given the learned projection, it needs $\mathcal{O}(nd)$ to perform the multiplication to derive the one dimension projected points. Note that the above time complexity is estimated for the upper bound since the as the recursive partition goes on, the size of the data points on each node reduces exponentially. The whole algorithm flow is summarized in Algorithm 1, where the leaf size threshold δ means the maximum number of data points allowed in each leaf node. It serves as an example of the stopping criterion of the tree construction procedure, meaning that we can also use other criteria (e.g., tree depth).

Algorithm 1. Adaptive Recursive Tree Partitioning (ART)

Require: Data set $\mathcal{X}$, Must-link set $\mathcal{M}$, Cannot-link set $\mathcal{C}$, Tradeoff parameter λ, Leaf size threshold δ

1: Let $\mathcal{T}$ be a hash table containing the tree nodes, and the keys of $\mathcal{T}$ are node indices. Initialize $\mathcal{T}\{0\} = \mathcal{X}$ and $\mathcal{R}_{old} = \{0\}, \mathcal{R}_{new} = \phi$.
2: **if** $\mathcal{R}_{old}$ is not empty **then**
3: **for** i in $\mathcal{R}_{old}$ **do**
4: Construct $\mathcal{J}_{\mathcal{S}}$ and $\mathcal{J}_{\mathcal{U}}$ using the data contained in $\mathcal{T}\{i\}$, and solve for the optimal $\mathbf{w}$
5: Project the data in $\mathcal{T}\{i\}$ onto $\mathbf{w}$. Let $\mathcal{T}\{2i+1\} = \{\mathbf{x} : \mathbf{w}^\top\mathbf{x} < 0\}$, and $\mathcal{T}\{2i+2\} = \{\mathbf{x} : \mathbf{w}^\top\mathbf{x} \geqslant 0\}$.
6: Check if $|\mathcal{T}\{2i+1\}| > \delta$, then add $2i+1$ to $\mathcal{R}_{new}$; if $|\mathcal{T}\{2i+2\}| > \delta$, then add $2i+2$ to $\mathcal{R}_{new}$
7: **end for**
8: $\mathcal{R}_{old} = \mathcal{R}_{new}, \mathcal{R}_{new} = \phi$
9: **else**
10: Output $\mathcal{T}$
11: **end if**

3.5. Kernelized tree construction

One limitation of the current formulation is that all the projections are assumed to be linear, which may not be optimal when the data distribution is nonlinear, and this is the common case for real world data. Kernel trick [11] is a popular choice to nonlinearize linear algorithms.

The basic idea of kernel trick is to first apply a nonlinear mapping $\phi : \mathbb{R}^d \to \mathbb{F}$ to map the data from the original d dimensional space to a high (could be infinite) dimensional feature space $\mathbb{F}$, such that the nonlinear problem in $\mathbb{R}^d$ becomes linear in $\mathbb{F}$. Let $\phi(\mathbf{x}_i) \in \mathbb{F}$ be the image of $\mathbf{x}_i$ after mapping, and $\mathbf{\Phi} = [\phi(x_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_n)]$. To kernelize the tree construction procedure we introduced above, we first project the data vectors to $\mathbb{F}$ via ϕ, and then get the projection direction $\boldsymbol{\omega} \in \mathbb{F}$ at each partitioning step by maximizing the kernelized objective of Eq. (8) as

$$\mathcal{J}_\phi = \boldsymbol{\omega}^\top\mathbf{\Phi}\mathbf{A}\mathbf{\Phi}^\top\boldsymbol{\omega} \tag{9}$$

According to the representer theorem, $\boldsymbol{\omega}$ can be represented as the linear combination of those data images as

$$\boldsymbol{\omega} = \sum_{i=1}^{n}\beta_i\phi(\mathbf{x}_i) \tag{10}$$

Let $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_n]^\top$, then Eq. (9) becomes

$$\mathcal{J}_\phi = \boldsymbol{\beta}^\top\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{A}\mathbf{\Phi}^\top\mathbf{\Phi}\boldsymbol{\beta} = \boldsymbol{\beta}^\top\mathbf{K}^{(n)}\mathbf{A}\mathbf{K}^{(n)}\boldsymbol{\beta} \tag{11}$$

where $\mathbf{K}^{(n)} \in \mathbb{R}^{n\times n}$ is the kernel matrix with $K^{(n)}_{ij} = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$. Therefore the exact form of ϕ is not required here, we only need to define a proper kernel function K.

By comparing Eqs. (9) and (8), we can see that the two objective will be the same if we treat $\mathbf{K}^{(k)}$ as the data matrix, i.e., each data vector $\mathbf{x}_i = \mathbf{K}^{(n)}_i \in \mathbb{R}^n$, where $\mathbf{K}_i$ represents the ith column of $\mathbf{K}$. One potential issue of such kernelized formulation is the *curse of dimensionality*, which happens when n is very large. This is a common problem for partitioning based methods because the higher the data dimensionality is, the more partition we need to get homogeneous data groups, and the deeper the indexing tree will be, so the retrieval speed using that tree will be longer.

To alleviate such problem, we propose a *sampling* based approach. We first sample $m \ll n$ landmark data points from the original data space $\mathbb{R}^d$, and construct a kernel matrix $\mathbf{K}^{(m)} \in \mathbb{R}^{m\times m}$. After eigenvalue decomposition, $\mathbf{K}^{(m)}$ can be expanded as

$$\mathbf{K}^{(m)} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top, \tag{12}$$

where $\mathbf{V} \in \mathbb{R}^{m\times m}$ is the eigenvector matrix and $\mathbf{\Lambda} \in \mathbb{R}^{m\times m}$ is the diagonal eigenvalue matrix. Let $\mathbf{K}^{(n,m)} \in \mathbb{R}^{n\times m}$ be the kernel matrix between the data and those landmarks, then according to [23], $\mathbf{K}^{(n)}$ can be approximated by

$$\mathbf{K}^{(n)} \approx \mathbf{K}^{(n,m)}\left(\mathbf{K}^{(m)}\right)^{-1}\left(\mathbf{K}^{(n,m)}\right)^\top = \mathbf{G}\mathbf{G}^\top \tag{13}$$

where $\mathbf{G} = \mathbf{K}^{(n,m)}\mathbf{V}\mathbf{\Lambda}^{-1/2} \in \mathbb{R}^{n\times m}$. Combining Eqs. (13) and (11), we can obtain

[1] If choosing $\mathcal{J}_{\mathcal{U}_{CS}}$ and $\mathcal{J}_{\mathcal{S}_{proj}}$, it is more convenient to form a minimization problem to derive the optimal projection.

$$\mathcal{J}_\phi = \boldsymbol{\beta}^\top \mathbf{G}\mathbf{G}^\top \mathbf{A}\mathbf{G}\mathbf{G}^\top \boldsymbol{\beta} = \boldsymbol{\eta}^\top \mathbf{G}^\top \mathbf{A}\mathbf{G}\boldsymbol{\eta} \quad (14)$$

where $\boldsymbol{\eta} = \mathbf{G}^\top \boldsymbol{\beta} \in \mathbb{R}^m$. Eqs. (14) and (8) are equivalent if we treat $\mathbf{G}^\top \in \mathbb{R}^{m \times n}$ as the data matrix. In this way, the data dimensionality is reduced to m. In order to make the approximation in Eq. (13) more accurate, we use the cluster sampling approach suggested in [26], where the landmarks are selected as the centers of the data clusters, obtained by simple K-means [2], in the original data space.

To summarize, we need three extra steps for constructing a kernelized tree: (1) Performing K-means and get the cluster centers, this will take $O(mnd)$ time. (2) Constructing $n \times n$ data kernel matrix $\mathbf{K}$, this will take $O(n^2)$ time. (3) Obtaining the approximation as in Eq. (13), this will take $O(m^2 n)$ time.

Table 1
Patient evaluation categories.

	Relevant	Irrelevant
Retrived	True Positives (TP)	False Positives (FP)
Nonretrieved	False Negatives (FN)	True Negatives (TN)

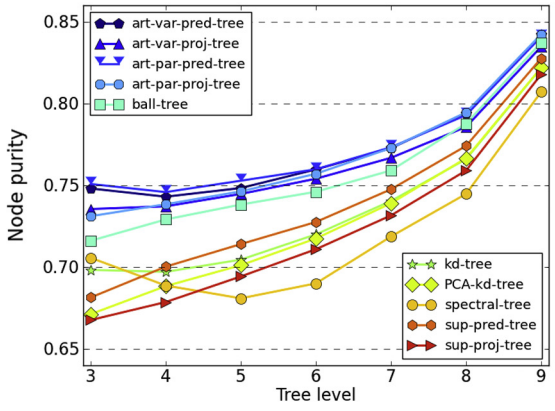
Algorithm 2. Kernelized ART (KART) partitioning

Require: Data set $\mathcal{X}$, Must-link set $\mathcal{M}$, Cannot-link set $\mathcal{C}$, Tradeoff parameter λ, Kernel function $\mathcal{K}$, number of landmarks m, Leaf size threshold δ

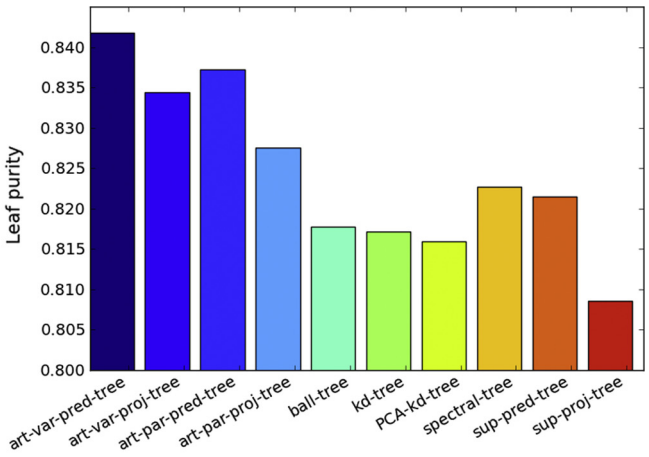
1: Cluster $\mathcal{X}$ into m clusters using K-means, and extract the cluster centers $\{\mathbf{c}_i\}_{i=1}^m$ as landmarks
2: Construct the kernel matrix $\mathbf{K}^{(m)}$ and $\mathbf{K}^{(n,m)}$ with $\mathbf{K}_{ij}^{(m)} = \mathcal{K}(\mathbf{c}_i, \mathbf{c}_j)$ and $\mathbf{K}_{ij}^{(n,m)} = \mathcal{K}(\mathbf{c}_i, \mathbf{x}_j)$
3: Perform complete eigenvalue decomposition on $\mathbf{K}^{(m)}$ as in Eq. (12)
4: Transform the original data matrix $\mathbf{X}$ into $\mathbf{G}$ as in Eq. (13)
5: Run Algorithm 1 on $\mathbf{G}$

4. Experiments on benchmark data

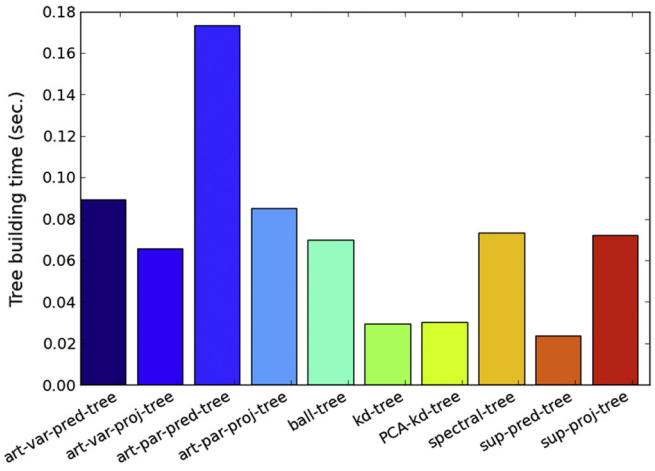
In this section we will present the experiments on applying our algorithm to two UCI benchmarks: i.e., **Breast Cancer** and **Diabetes** datasets, whose major information are summarized in the following subsection. The purpose of the empirical evaluation is to answer the following questions:



(a) Node purity

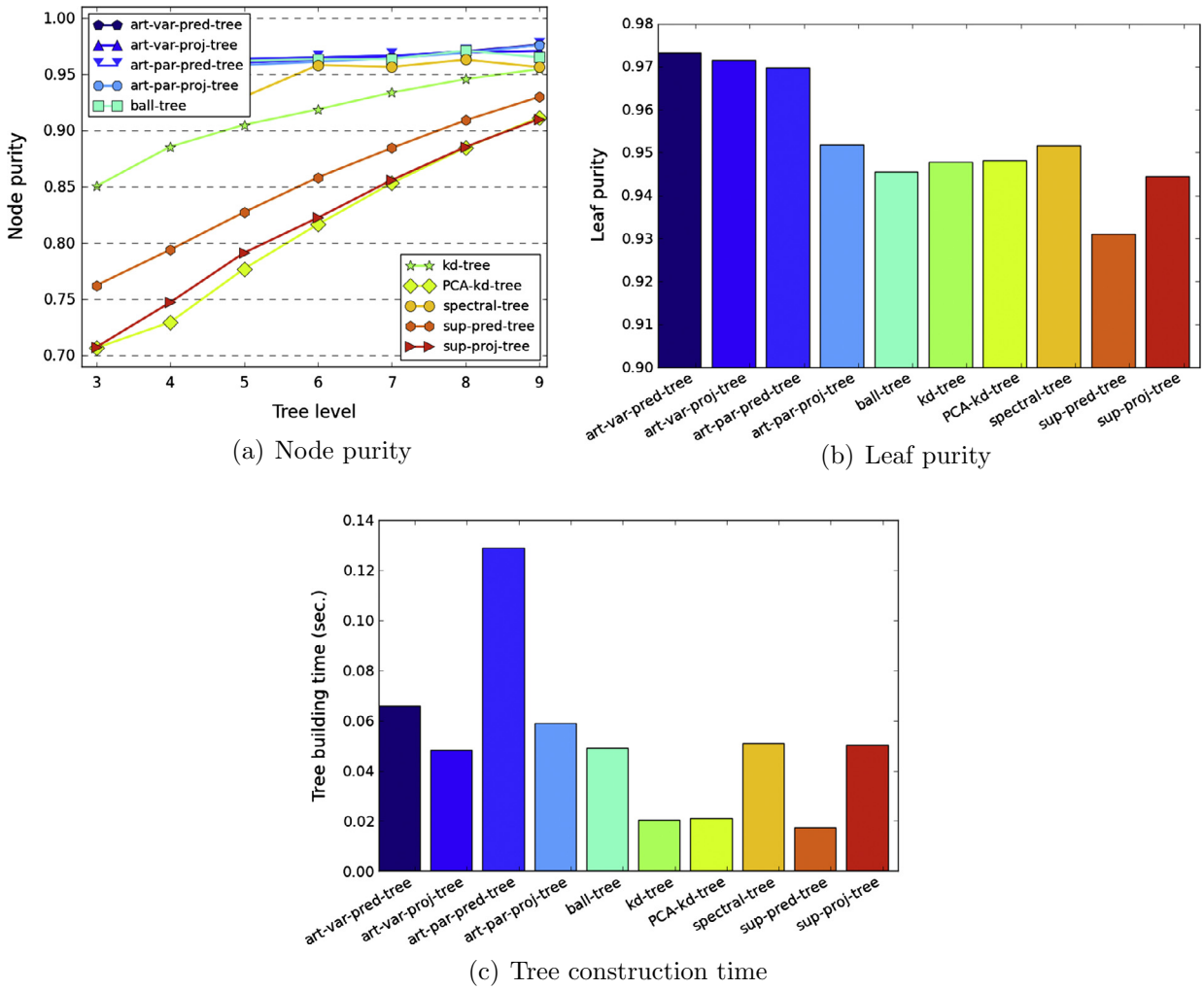


(b) Leaf purity



(c) Tree construction time

Fig. 1. Average statistics of the trees constructed by different methods on Diabetes data set with the maximum number of data objects in each leaf node set to 5. (a) Shows the average node purity on each level of tree nodes. (b) Shows the average purity of the leaf nodes. (c) Shows the average tree construction time.



(a) Node purity

(b) Leaf purity

(c) Tree construction time

Fig. 2. Average statistics of the trees constructed by different methods on Breast Cancer data set with the maximum number of data objects in each leaf node set to 5. (a) Shows the average node purity on each level of tree nodes. (b) Shows the average purity of the leaf nodes. (c) Shows the average tree construction time.

- How **efficient** we can retrieve the patients with the proposed approach?
- How **accurate** we can retrieve the patients with the proposed approach?

All experiments are conducted on a MAC machine with OS version 10.7.5, 2.2 GHz CPU and 12 GB RAM. The development environment is Python 2.7 with numpy and scipy.

4.1. Data set information

The **Breast Cancer (Diagnostic)** data set contains 569 data vectors with dimensionality 30.[2] Each data vector is computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe the characteristics of the cell nuclei present in the image. All the samples will be categorized as either *malignant* (positive) or *benign* (negative).

The **Pima Indians Diabetes** data set contains 768 patients which are all females at least 21 years old of Pima Indian heritage.[3] Each patient is represented by a 8 dimensional vector. Finally each patient will be classified as either diabetic patients (positive) or not (negative). Apparently, both datasets can be treated as binary classification problems.

[2] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).
[3] http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes.

4.2. Evaluation metrics

We use the following metrics to evaluate the performance of various indexing approaches.

4.2.1. Node purity

As we discussed earlier, the tree-based indexing algorithms recursively partition the data set to construct a tree structure until there is only a limited number of data points contained in each leaf node. We use *node purity* to measure the label consistency of the data points contained in individual tree nodes. Mathematically,

$$Purity(e) = \max_c |\mathbf{x}_i \in e, l_i = c|/|e| \quad (15)$$

where e is the set of data points in a node on the indexing tree and $|e|$ is its cardinality. $|\mathbf{x}_i \in e, l_i = c|$ indicates the number of data points in e with the same class label c. In particular, the computed node purity for leaf nodes is called *leaf purity*, which directly reflects the search accuracy.

4.2.2. Retrieval statistics

Before computing the evaluation metrics, we first construct a contingency table shown in Table 1 according to the labels of the retrieved and the query patients, where *relevant* means the retrieved patient has the same label as the query patient, otherwise the retrieved patients is *irrelevant*. Then we adopt the following four metrics to evaluate the performance of PSF:

- *Precision*, denoted as *P*, is the fraction of retrieved patients that are relevant.

$$P = \frac{TP}{TP + FP} \tag{16}$$

- *Recall*, denoted as *R*, is the fraction of relevant patients that are retrieved.

$$R = \frac{TP}{TP + FN} \tag{17}$$

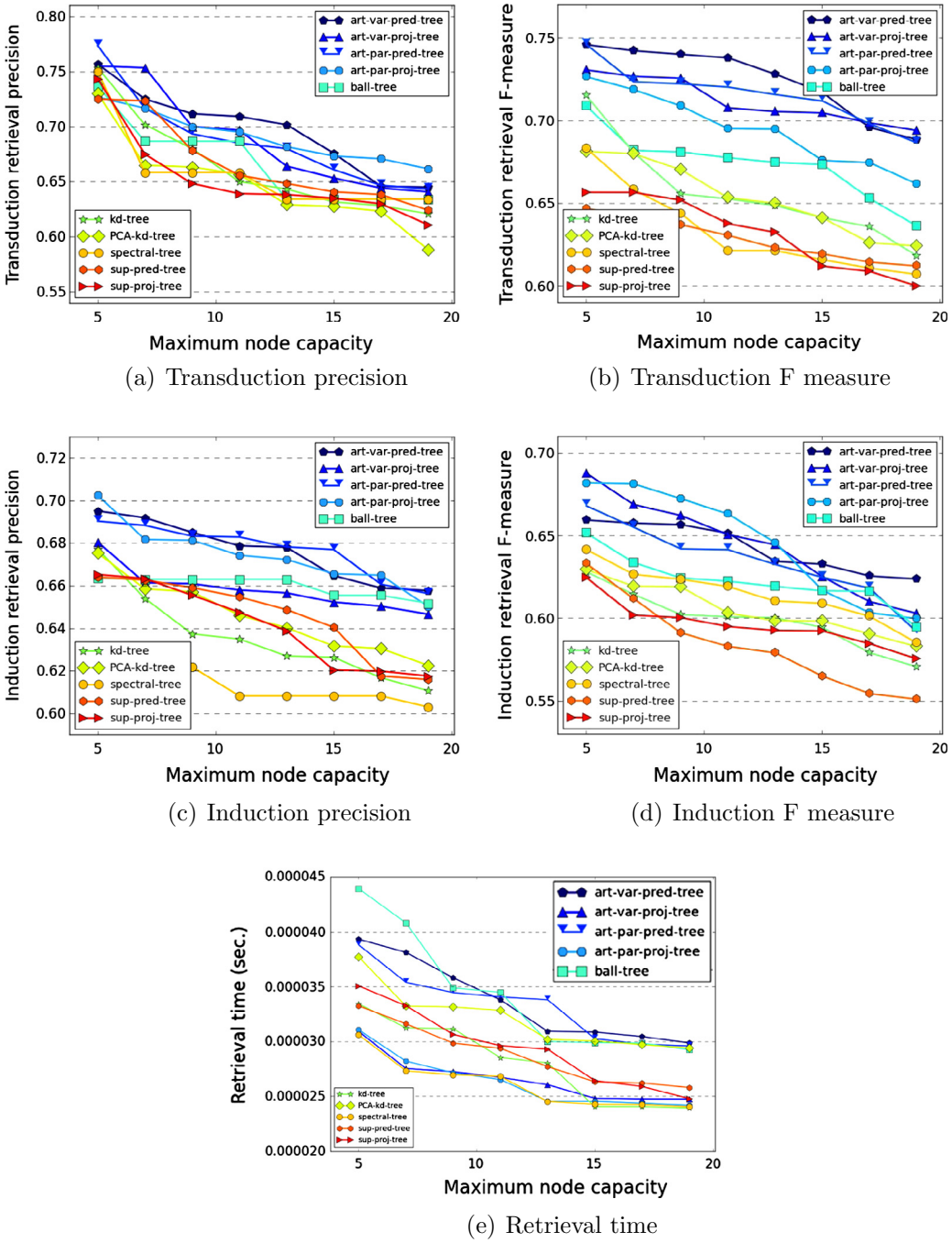
- *F-measure*, denoted as *F*, is the weighted harmonic mean of precision and recall.

$$F = \frac{2PR}{P + R} \tag{18}$$

4.3. Algorithms for comparison

We implemented the following algorithms for comparison:

- **art-var-pred-tree:** This refers to our ART strategy with $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ constructed by the variance maximization based method in Section 3.1.1 and $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ constructed from the prediction perspective in Section 3.2.2.



(a) Transduction precision

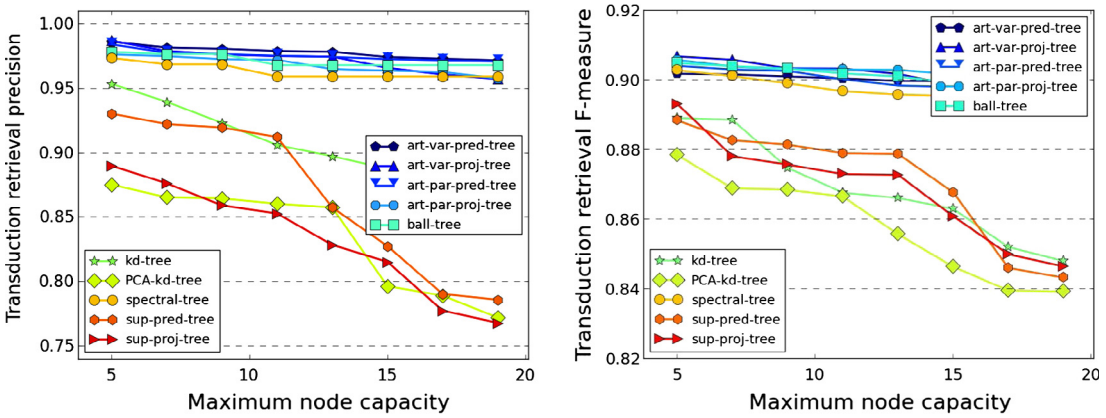
(b) Transduction F measure

(c) Induction precision

(d) Induction F measure

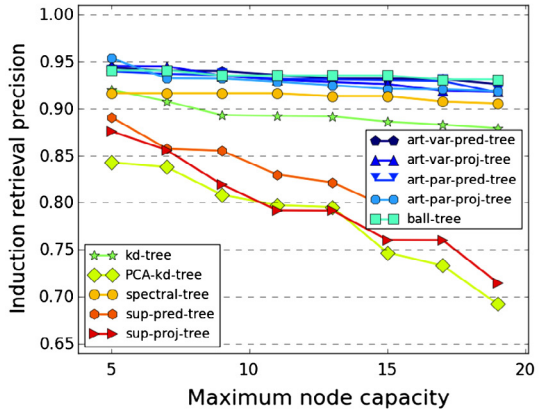
(e) Retrieval time

Fig. 3. Average retrieval statistics of different methods on Diabetes data set with different maximum number of data objects in each leaf node. (a) Shows the transduction precision. (b) Shows the transduction F measure. (c) Shows the induction precision. (d) Shows the induction F measure. (e) Shows the retrieval time.

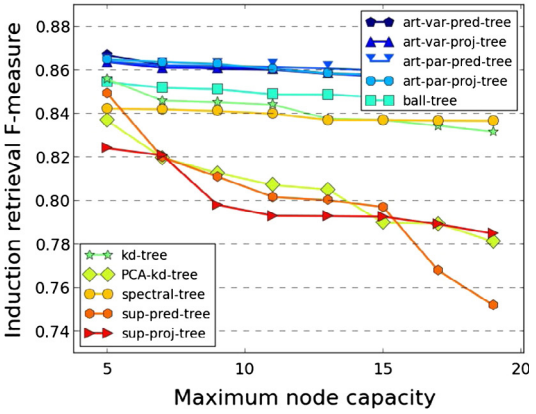


(a) Transduction precision

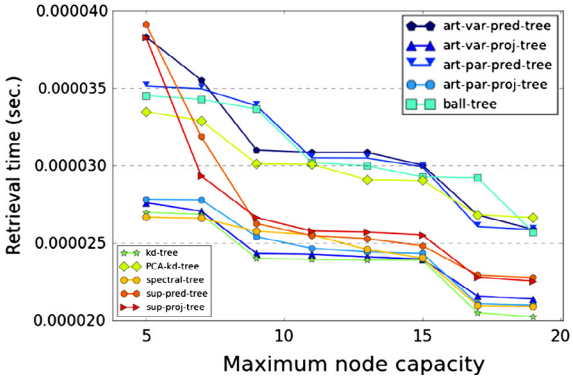
(b) Transduction F measure



(c) Induction precision



(d) Induction F measure



(e) Retrieval time

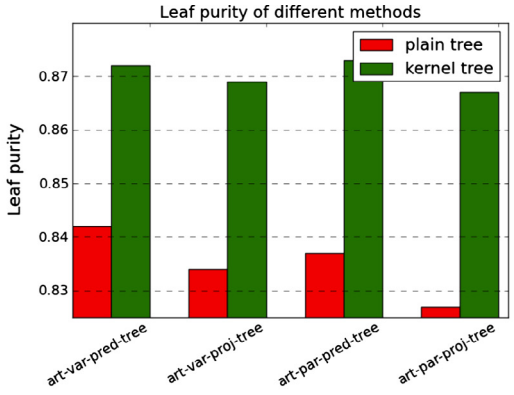
Fig. 4. Average retrieval statistics of different methods on Breast Cancer data set with different maximum number of data objects in each leaf node. (a) Shows the transduction precision. (b) Shows the transduction F measure. (c) Shows the induction precision. (d) Shows the induction F measure. (e) Shows the retrieval time.

- **art-var-proj-tree:** This refers to our ART strategy with $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ constructed by the variance maximization based method in Section 3.1.1 and $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ constructed from the projection perspective in Section 3.2.1.
- **art-par-pred-tree:** This refers to our ART strategy with $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ constructed by the cluster partition based method in Section 3.1.2 and $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ constructed from the prediction perspective in Section 3.2.2.
- **art-par-proj-tree:** This refers to our ART strategy with $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ constructed by the cluster partition based method in Section 3.1.2 and $\mathcal{J}_{\mathcal{S}}(\mathbf{w})$ constructed from the projection perspective in Section 3.2.1.
- **kd-tree:** We used the implementation in `scikit-learn`.[4]
- **PCA-kd-tree:** This is the method we first transform the data using principal component analysis, then perform kd-tree on top of that.
- **ball-tree:** We used the implementation in `scikit-learn`.
- **spectral-tree:** This corresponds to the tree indexing method where at each internal node we only find a direction that minimizes $\mathcal{J}_{\mathcal{U}}(\mathbf{w})$ constructed by cluster partition based method in Section 3.1.2.

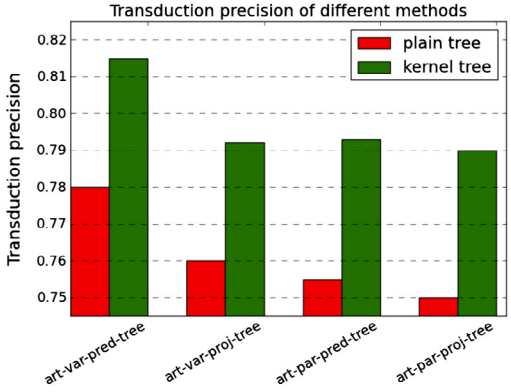
[4] http://scikit-learn.org/stable/.

- **sup-pred-tree:** This is the method we first project the data on the directions obtained by maximizing $\mathcal{J}_{\mathcal{S}_{pred}}$ in Section 3.2.2, then perform kd-tree on top of that.
- **sup-proj-tree:** This is the method we first project the data on the directions obtained by maximizing $\mathcal{J}_{\mathcal{S}_{proj}}$ in Section 3.2.1, then perform kd-tree on top of that.

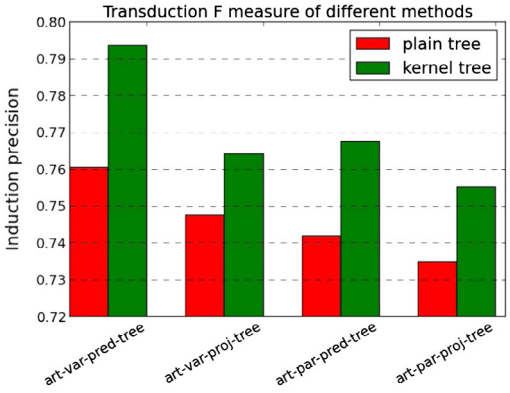
For all methods, if not explicitly presented, the maximum number data points allowed in a leaf node is set to 5. For all methods involve supervision information, we randomly label 10% of the data. The coefficient λ in `ART`-series methods is determined using 5-fold cross validation with average leaf purity from $\{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$. In `spectral-tree`, the pairwise data similarity is computed using Gaussian function with the bandwidth set as the average Euclidean distance between each pair of samples. For all trees, the index of root node is set as level 0 and the index of level nodes keep increasing as the tree goes deeper. The reported results are computed from 100 independent and random runs, where for each run we use a different randomization seed for labeling the data.



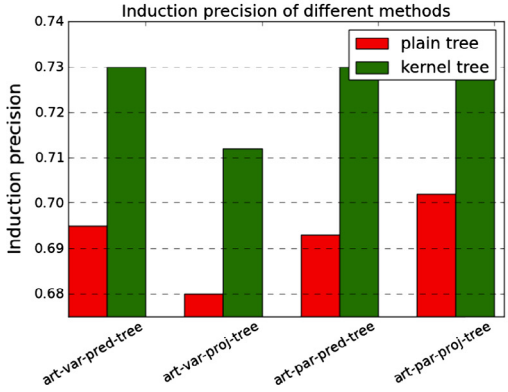
(a) Leaf purity



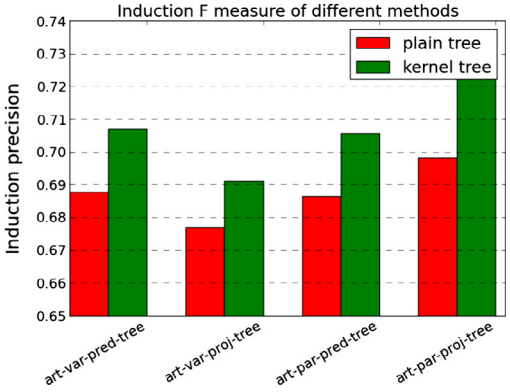
(b) Transduction precision



(c) Transduction F measure



(d) Induction precision

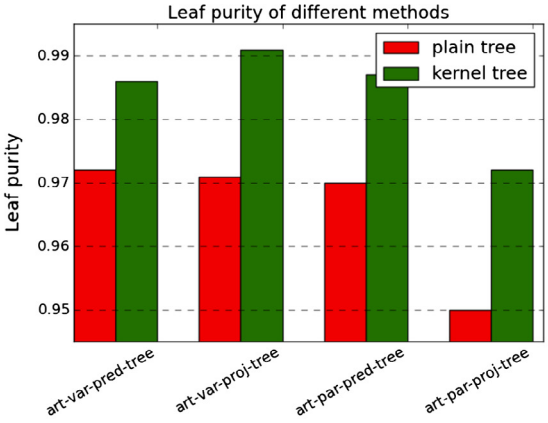


(e) Induction F measure

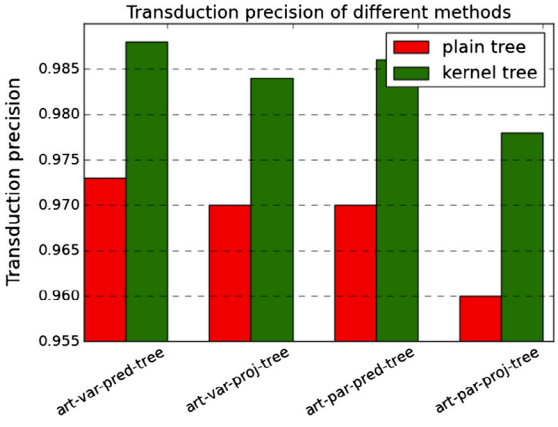
Fig. 5. Performance comparison on Diabetes data set of different `ART`-series tree construction methods with or without kernels. The leaf size threshold δ is set to 5. The number of landmarks is set to 50. (a) Shows the leaf purity. (b) Shows the transduction precision. (c) Shows the transduction F measure. (d) Shows the induction precision. (e) Shows the induction F measure. The performance values in all figures are averaged over 100 independent runs.

4.4. Results

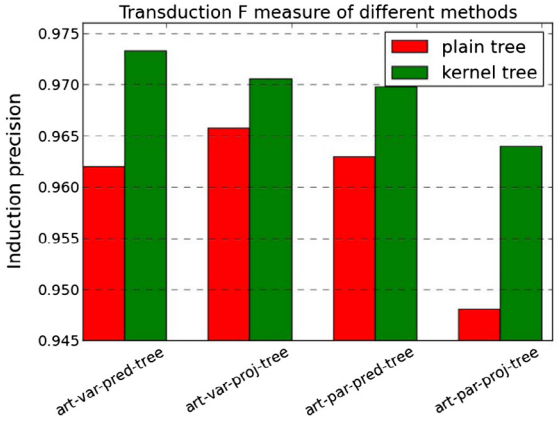
We conducted two sets of experiments to validate the effectiveness and efficiency of the proposed ART indexing schemes. In the first set of experiments, we evaluated the indexing tree statistics, including the average node purity, average leaf purity and average tree construction time. In the second set of experiments, we evaluated the query retrieval statistics using those learned indexing trees, including average transduction precision, average induction precision and average retrieval time. Here **transduction** means that the query is also coming from the data that used to construct the indexing tree. **Induction** means that the query is not in the data that used to construct the tree. For induction, we randomly sample 90% of the whole data set for building the indexing tree, and the queries are selected from the rest 10% data. One reason that we choose the 90:10 training/testing ratio here is that the



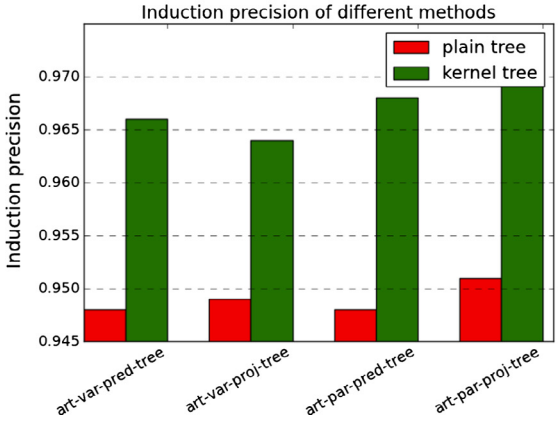
(a) Leaf purity



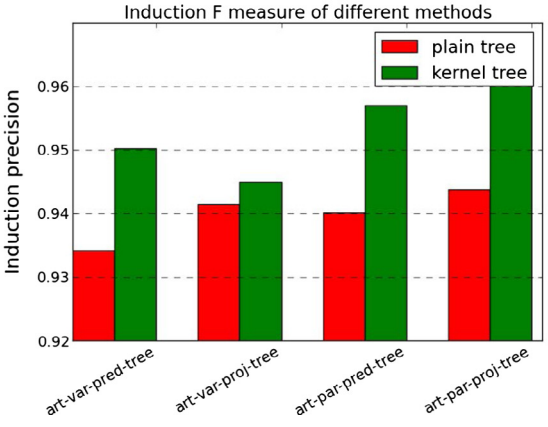
(b) Transduction precision



(c) Transduction F measure

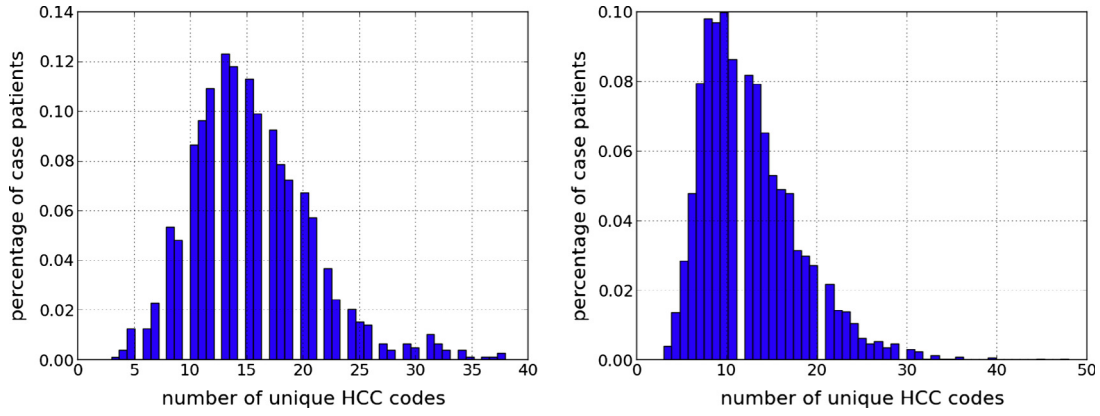


(d) Induction precision



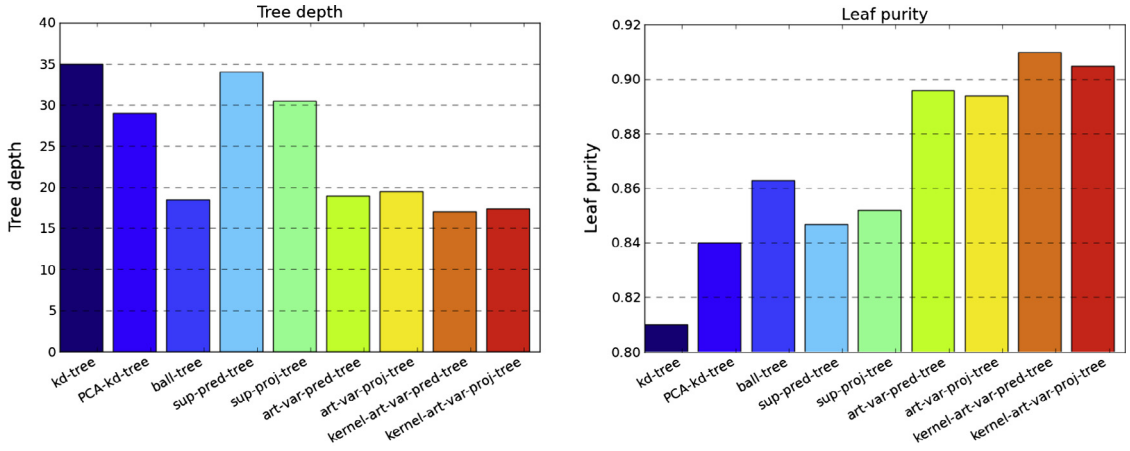
(e) Induction F measure

Fig. 6. Performance comparison on Breast Cancer data set of different ART-series tree construction methods with or without kernels. The leaf size threshold δ is set to 5. The number of landmarks is set to 50. (a) Shows the leaf purity. (b) Shows the transduction precision. (c) Shows the transduction F measure. (d) Shows the induction precision. (e) Shows the induction F measure. The performance values in all figures are averaged over 100 independent runs.

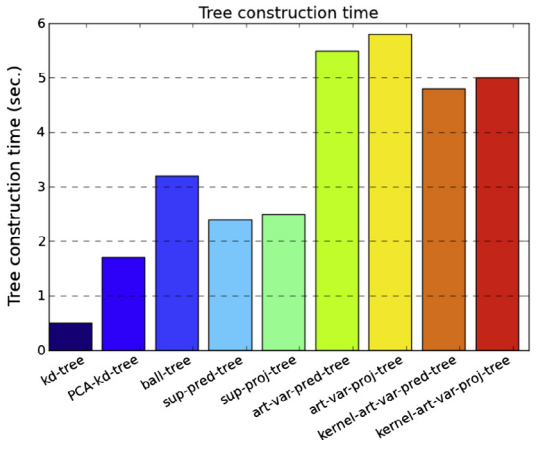


(a) Case (b) Control

Fig. 7. The histogram of the unique HCC codes appeared in case and control patients.



(a) Tree Depth (b) Leaf purity



(c) Tree construction time

Fig. 8. Average statistics of the trees constructed by different methods on the real world data set with the maximum number of data objects in each leaf node set to 10. (a) Shows the average node purity on each level of tree nodes. (b) Shows the average purity of the leaf nodes. (c) Shows the average tree construction time. We set the number of landmarks to 100 for the two kernel tree construction methods.

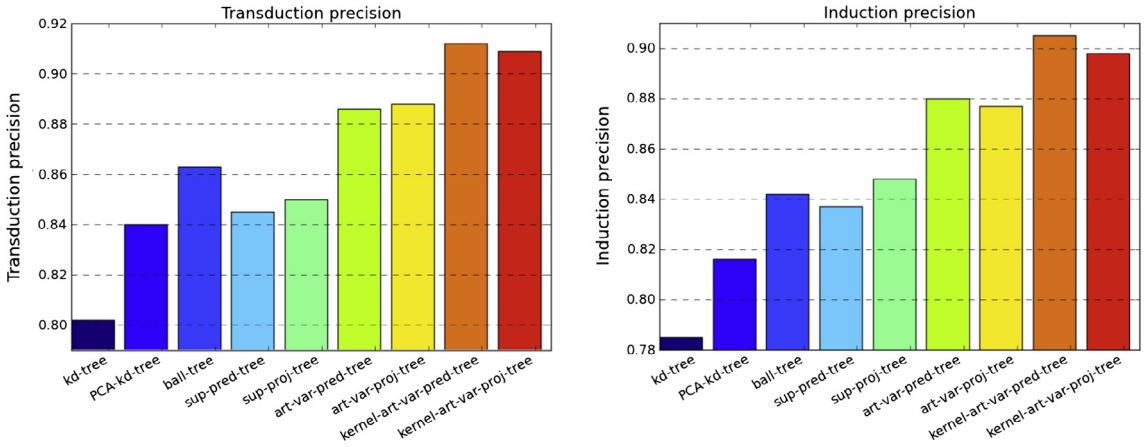
benchmark data we used here are relative small, we need to make sure the training data set is sufficiently large to learn the indexing tree. Note that the more labeled data we have, the more accurate indexing tree we can built, we can expect better retrieval performance. However, the retrieval time should not be affected because the complexity of the tree is not related to the labeled data size.

Figs. 1 and 2 show the statistics of the constructed indexing trees using Diabetes and Breast Cancer data sets. For subfigure (a) in both figures, the horizontal corresponds to the tree levels, and the vertical axis represents the averaged node purity of each specific level. From the figures we can see a clear increasing trends of all curves. This is because with the tree going deeper, the data will be partitioned into small yet purer chunks. Another observation here is that the art series methods can achieve purer nodes compared to other competitors. Subfigure (b) in both figures show the average purity of the leaf nodes on the indexing tree for both data sets, which indicate that art series methods can achieve higher leaf purity compared to other methods, so that the retrieved nearest neighbors will be more accurate. Subfigure (c) show the tree construction time for different approaches. Art series methods take more time to construct the indexing tree due to the additional time cost for computing the optimal projection and partitioning. However, since the tree construction process can be done offline and it does not affect the online search efficiency.

Figs. 3 and 4 show the evaluation results of retrieval statistics on Diabetes and Breast Cancer data sets. Subfigure (a)/(b) in both figures show the average transduction/induction retrieval precision over 100 independent runs, from which we can see that art series methods and ball tree algorithm perform better than other methods. Subfigure (c) in both figures show the average retrieval time, which suggest that the retrieval time of all those methods are comparable to each other, as the time granularity on the vertical axes is very small.

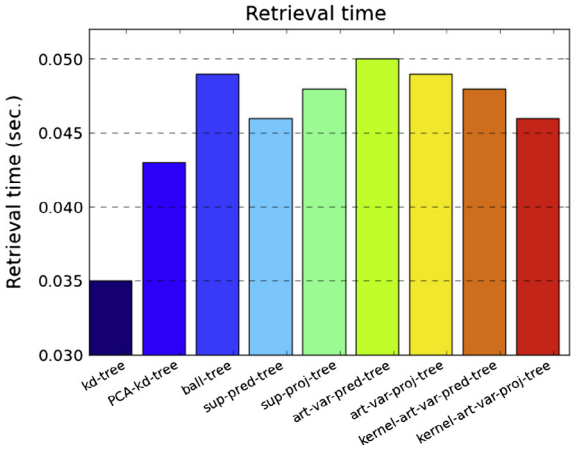
From those figures we can observe that generally the *art* series methods perform much better than those baseline methods, and some unsupervised baseline methods, such as *spectral-tree*, can achieve considerable performance as art methods. This is because those methods capture the data structure very well, which makes the supervision information not that important. This also explains why those supervised methods (*sup-pred-tree* and *sup-proj-tree*) do not perform well – because the number of labeled data is too small. However, these supervision are still helpful, that's why those art series methods can beat unsupervised methods in most of the scenarios.

We also tested how much the kernel trick will help. We first run standard K-means with randomly initialization to the data and set the number of clusters to be 50. We set this number as a consideration of the tradeoff between complexity and accuracy. On one hand, we do not want the number of clusters to be too large such that the computational complexity would be huge. On the other hand, we do not want the number of clusters to be too small so that the quality of the constructed tree is bad. We used Gaussian kernel with width tuned via 5-fold cross validation on the training data set from the grid $4^{[-4:4]}$. We compared the leaf purity, transduction



(a) Transduction precision

(b) Induction precision



(c) Retrieval time

Fig. 9. Average retrieval statistics of different methods on the real world data set with the maximum number of data objects in each leaf node set to 10. (a) Shows the transduction precision. (b) Shows the induction precision. (c) Shows the retrieval time. We set the number of landmarks to 100 for the two kernel tree construction methods.

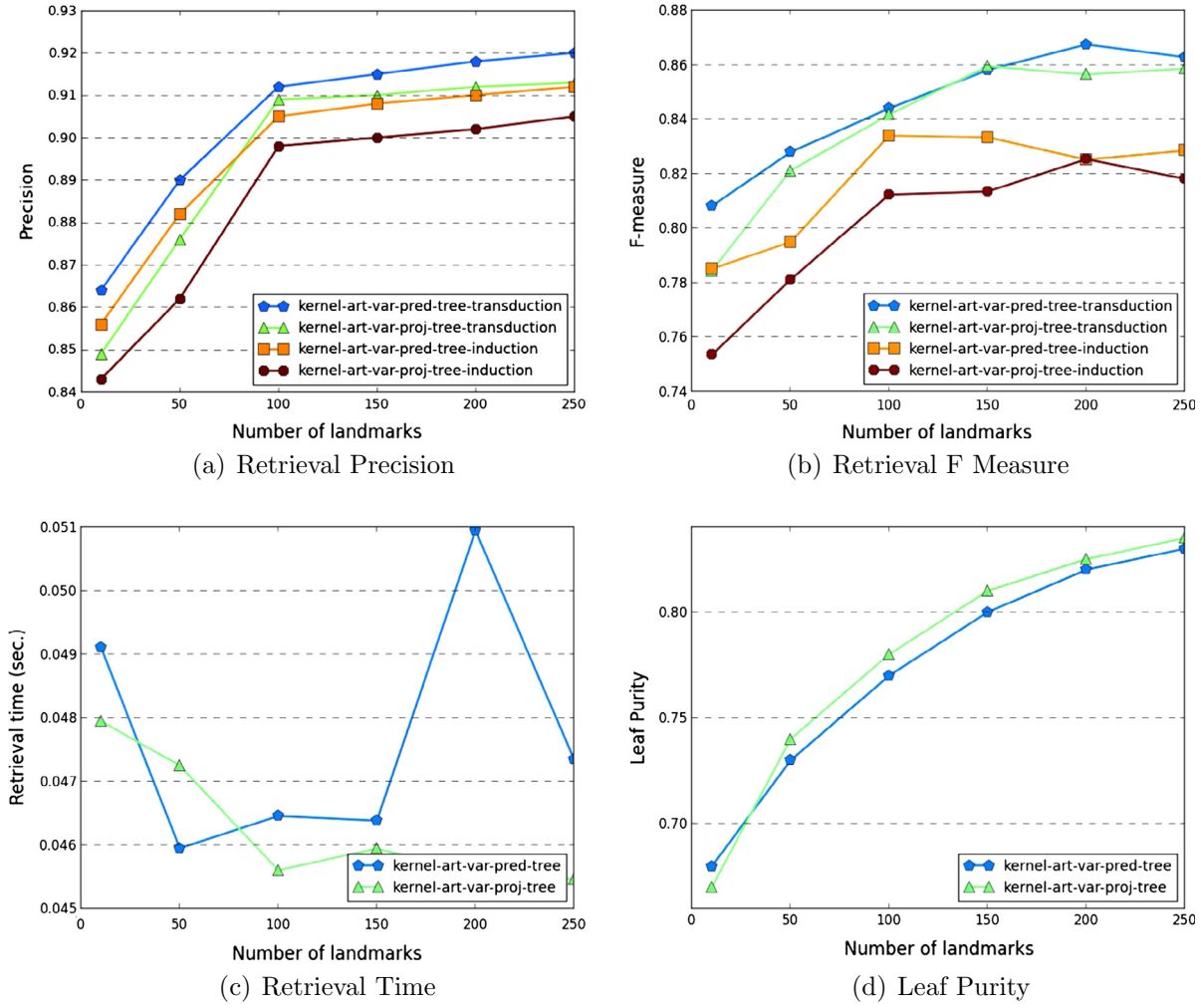
precision and induction precision of the ART series methods and their kernelized counterparts on Diabetes and Breast Cancer data sets, and the results are shown on Figs. 5 and 6. From those figures we can see that the performance of the ART series methods can be greatly boosted after the kernel trick is incorporated.

5. A real world case study

In this section we will present a real world case study using our ART series methods. The patient data warehouse we have access to is the 7-year longitudinal EMR data of 31,340 patients focusing on *Congestive Heart Failure* (CHF) study. The goal is to predict the risk of CHF, i.e., based on the patient's current records, predicting whether he/she will have CHF within 6 months [24]. In our empirical study, we anchor all patient records at the *operational criteria date*, which is the diagnosis date for CHF case patients, and match date for the control patients based on the clinical trial design. We collected the *Hierarchical Condition Categories* (HCC) codes appeared within 18 months to 6 months before the operational criteria dates as features. There are a total of 195 unique HCC codes. Fig. 7 shows the histogram of the unique HCC codes appeared in case and control patients, which shows the number of different disease conditions that the case and control patients have. During the construction of the indexing tree for similar patient search, we fix the maximum number of patients within each node to be 10. For kernelized tree construction, we set the number of landmarks to 100. All other parameters are set in the same way as last section. Since this dataset has tens of thousands of samples, it is infeasible to compute the pairwise similarity matrix. Therefore, we exclude `art-par-proj-tree`,`art-par-pred-tree` and `spectral-tree` in the experiments because they all require to calculate the pairwise similarity matrix. During the evaluation, we randomly sample 80% of the data for training, and the rest 20% for testing, and the performance reported below are averaged over 50 independent runs.

Fig. 8 shows the averaged (over 50 independent runs) tree statistics of different methods, and Fig. 9 shows the averaged (over 50 independent runs) retrieval statistics of different methods. From these results we can observe that:

- The depth of the trees constructed by ART methods is shallower, but the indexing is more accurate since the leaf purity of ART trees is much greater. However, it needs more time to construct ART trees.
- Kernel trick can make the constructed tree more precise with similar construction time. Note here we exclude the time for computing the kernels.
- Similar patient retrieval using the ART tree is more precise, and the retrieval time is similar as that using other types of trees.



(a) Retrieval Precision (b) Retrieval F Measure

(c) Retrieval Time (d) Leaf Purity

Fig. 10. Comparison of retrieval precision and F measure of transduction and induction, as well as retrieval time and resultant leaf purity of different methods with varying number of selected landmarks.

We also tested the performance of kernalized tree construction methods with different number of landmarks. The transduction and induction precision results are shown in Fig. 10, where we vary the landmark number from 10 to 250. From the figure we can observe a clear increasing trend with more landmark points. Moreover, there is an elbow point on those curves when the number of landmark points equals 100, which suggests the performance improvement over 100 becomes slower, thus 100 is a good choice for kernelized tree construction.

6. Discussion

We proposed a general semi-supervised patient indexing framework in this paper. Our framework is fairly general in the sense that its application will not be restricted in just patient data or medical applications, but also other data analytics problems as long as the data can be represented as vectors. From the empirical study results on both benchmark data and real world case study we can have the following observations:

- The supervised methods may perform poorly when the amount of supervision information is very small.
- In the scenarios where not enough supervision is available, unsupervised methods could perform very well as long as it can explore the data characteristics properly.
- In general the optimal methodology is combining both supervised and unsupervised approaches, which is just the design principle for art series methods.
- There are only subtle differences among the four different instantiations of the art series methods proposed in this paper, so it is hard to tell which one is better. However, the goal of this paper is to propose a general semi-supervised approach for building smart indexing structure. In practice, the user can design specific $\mathcal{J}_S(\mathbf{w})$ and $\mathcal{J}_U(\mathbf{w})$ terms according to their knowledge.

7. Conclusions

In this paper, we propose an Adaptive Semi-Supervised Recursive Tree Partitioning (ART) framework for large scale patient indexing and search. The tree structure is recursively built based on solving an optimization problem whose objective is composed of two terms. One is a supervised term enforcing such indexing should be comply with the prior supervision knowledge in terms of pairwise constraints. The other is an unsupervised term exploring the geometric structure of the patient vectors. We also provide four different instantiations of the ART framework. With such a framework, we can rapidly and accurately retrieve the similar patients. Empirical validation over both benchmark and real world patient data clearly show that the proposed method achieves much higher performance, compared to several state-of-the-art supervised or unsupervised tree indexing approaches, while the retrieval time of our methods is still comparable to those baseline methods.

References

[1] Bentley J. Multidimensional binary search trees used for associative searching. Commun. ACM 1975;18(9):517.
[2] Elkan C. Using the triangle inequality to accelerate k-means. In: Proceedings of international conference on machine learning; 2003. p. 147–53.
[3] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. In: Proc. of 25th international conference on very large data base; 1999. p. 518–29.
[4] Horn RA, Johnson CR. Matrix analysis. Cambridge university press; 2012.
[5] Jolliffe IT. Principal component analysis. 2nd ed. Springer; 2002.
[6] Kulis B, Darrell T. Learning to hash with binary reconstructive embeddings. Adv Neural Inf Process Syst 2009;20:1042–50.
[7] Liu T, Moore AW, Gray A, Yang K. An investigation of practical approximate nearest neighbor algorithms. In: Saul LK, Weiss Y, Bottou L, editors, Advances in neural information processing systems. vol. 17; 2005. p. 825–32.
[8] Mayer-Schönberger V, Cukier K. Big data: a revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt; 2013.
[9] Ng A, Jordan M, Weiss Y, et al. On spectral clustering: analysis and an algorithm. Adv Neural Inf Process Syst 2002:849–56.
[10] Omohundro S. Efficient algorithms with neural network behavior. Complex Syst 1987;1(2):273–347.
[11] Schölkopf B, Smola AJ. Learning with kernels: support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning). 1st ed. The MIT Press; 2001.
[12] Shakhnarovich G, Darrell T, Indyk P. Nearest-neighbor methods in learning and vision: theory and practice. MIT Press; 2006.
[13] Su X, Khoshgoftaar TM. A survey of collaborative filtering techniques. Adv Artif Intell 2009;2009:4.
[14] Sun J, Sow DM, Hu J, Ebadollahi S. Localized supervised metric learning on temporal physiological data. In: The 20th international conference on pattern recognition; 2010. p. 4149–52.
[15] Tao D, Li X, Wu X, Maybank SJ. General tensor discriminant analysis and Gabor features for gait recognition. IEEE Trans Pattern Anal Mach Intell 2007;29(10):1700–15.
[16] Tao D, Li X, Wu X, Maybank SJ. Geometric mean for subspace selection. IEEE Trans Pattern Anal Mach Intell 2009;31(2):260–74.
[17] Tao D, Tang X, Li X, Wu X. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Trans Pattern Anal Mach Intell 2006;28(7):1088–99.
[18] Uhlmann J. Satisfying general proximity/similarity queries with metric trees. Inf. Process. Lett. 1991;40(4):175–9.
[19] Wang F, Sun J, Ebadollahi S. Integrating distance metrics learned from multiple experts and its application in inter-patient similarity assessment. In: The 11th SDM; 2011. p. 59–70.
[20] Wang F, Sun J, Hu J, Ebadollahi S. Integrating distance metrics learned from multiple experts and its application in inter-patient similarity assessment. In: The 11th SIAM data mining conference; 2011. p. 944–55.
[21] Wang J, Kumar S, Chang S. Semi-supervised hashing for large scale search. IEEE Trans Pattern Anal Mach Intell 2012.
[22] Wang J, Kumar S, Chang S-F. Sequential projection learning for hashing with compact codes. In: Proceedings of the 27th international conference on machine learning; 2010. p. 1127–34.
[23] Williams C, Seeger M. Using the nyström method to speed up kernel machines. Adv Neural Inf Process Syst 2001;13:682–8.
[24] Wu J, Roy J, Stewart WF. Prediction modeling using ehr data: challenges, strategies, and a comparison of machine learning approaches. Med Care. 2010;48(6 Suppl):S106–13.
[25] Xu C, Tao D, Xu C. Large-margin multi-view information bottleneck. IEEE Trans Pattern Anal Mach Intell 2014;36(8):1559–72.
[26] Zhang K, Tsang IW, Kwok JT. Improved nyström low-rank approximation and error analysis. In: Proceedings of international conference on machine learning; 2008, p. 1232–9.